CORE JAVA MODULE 5 NOTES

01 July 2025 14:29

➢ Module 5: Methods

No.	Topic
5.1	Creating & Calling Methods
5.2	Method Parameters, Return Types
5.3	Method Overloading

⇒ Module 5.1 – Creating & Calling Methods

✧ What is a Method?

A method is a block of code that performs a specific task and can be called multiple times.

Real-Life Analogy

Imagine a **vending machine**:

- Press button A → You get Coke
- Press button B → You get Chips

Each button is like a **method**. Once defined, you just call it to reuse its function.

Write once, use many times

Why Use Methods?

Purpose	Benefit
Reusability	Call again and again without rewriting code
Modularity	Break down complex problems into small tasks
Readability	Cleaner and easier-to-understand code
Maintenance	Easy to update or fix code in one place

◆ Basic Syntax of a Method

```
returnType methodName(parameters) {
    // code block
    return value;
}
```

1. Method Without Parameters & Return

```
public static void greet() {
    System.out.println("Hello Student!");
}
```

Call it: greet();

Behind the Scenes – Stack Memory

Every method call is pushed onto the **stack**. When the method finishes, it is **popped off** the stack.

main() → calls greet()
→ greet() finishes → returns to main()

Example Program

```
public class MethodDemo {
    public static void main(String[] args) {
        greet();
    }

    public static void greet() {
        System.out.println("Welcome to Java");
    }

}
```

Common Mistakes

Mistake	Fix
◆ Forgetting to call the method	◆ Just defining it won't execute — you must call it
◆ Wrong return type	◆ Return must match method's declared return type
◆ Not using static inside main	◆ Use static methods or create object to call
◆ Forgetting return statement	◆ For non-void return types, always return something
◆ Parameter mismatch	◆ Arguments must match the method's parameter list

Assignment Practice

1. Write a method to return the cube of a number.
2. Write a method to find the largest of 3 numbers.
3. Create a method to calculate factorial of a number.
4. Write a method to check if a number is even or odd.
5. Create a method to print a multiplication table for a number.

⇒ Module 5.2 – Method Parameters & Return Types

✧ What Are Parameters & Return Types?

Methods can **accept inputs (parameters)** and **give back outputs (return types)** — just like a vending machine that takes your button press and gives you snacks 🥤.

Real-Life Analogy

Scenario	Java Equivalent
You give ₹10 and press "A"	Parameters (int money)
The machine gives you Coke	Return value (String drink)
No button pressed	No parameters
Machine beeps but gives nothing	void return type

✧ Why Use Parameters & Return Types?

Concept	Purpose
Parameters	Provide method with input values
Return Type	Get results back from the method

Method Syntax Recap

```
returnType methodName(type1 param1, type2 param2) {
```

```
    // code
    return value;  // if not void
}
```

Example 1: Method with Parameters & Return

```
public static int add(int a, int b) {
    return a + b;
}
```

Call Method :

```
int result = add(10, 20);
System.out.println(result);  // 30
```

Example 2: Method with No Parameters, No Return

```
public static void greet() {
    System.out.println("Good Morning!");
}
```

Call: greet();

Example 3: Method With Parameters, No Return

```
public static void printSquare(int n) {
    System.out.println("Square: " + (n * n));
}
```

Example 4: Method With Return, No Parameters

```
public static String getWelcomeMessage() {
    return "Welcome to Java!";
}
```

Return Types: What You Can Return?

Return Type	Example
◆ int	◆ return 10;
◆ double	◆ return 99.5;
◆ String	◆ return "Hello";
◆ boolean	◆ return true;
◆ void	◆ No return statement

✧ Module 5.3 – Method Overloading

What is Method Overloading?

- Method Overloading means **defining multiple methods with the same name** but **different parameters** (number, type, or order).
- Java decides which one to run **at compile-time**, based on the arguments.

✧ Real-Life Analogy: Withdraw Money (Overloading)

Real-world situation:

You go to a bank and want to withdraw money. There are **multiple ways** to do it:

Withdrawal Method	What You Provide	System Action
ATM	◆ Amount only	◆ Withdraw instantly from your default account
Cheque	◆ Amount + Cheque Number	◆ Bank verifies cheque, then withdraws
In-Person (Branch)	◆ Amount + ID Proof + Signature	◆ Bank staff verifies your ID & signature, then withdraws

➢ Even though the **action is same** (withdraw money), the **way you request it changes** based on parameters.

Program :

```
public class Bank {

    // 1. Withdraw using ATM
    public void withdraw(int amount) {
        System.out.println("Withdrawing ₹" + amount + " via ATM.");
    }

    // 2. Withdraw using Cheque
    public void withdraw(int amount, String chequeNumber) {
        System.out.println("Withdrawing ₹" + amount + " via Cheque #" + chequeNumber);
    }

    // 3. Withdraw at Bank Counter
    public void withdraw(int amount, String idProof, String signature) {
        System.out.println("Withdrawing ₹" + amount + " at Bank Counter.");
        System.out.println("Verified ID: " + idProof + ", Signature: " + signature);
    }
}


public class BankApp {
    public static void main(String[] args) {
        Bank bank = new Bank();

        bank.withdraw(1000);                              // ATM
        bank.withdraw(5000, "CHQ123456");                 // Cheque
        bank.withdraw(10000, "Aadhar", "Jatin_Signature");    // Bank Counter
    }
}
```


- Java doesn't care about what the method does, it only checks the **method signature** (name + parameters). So you can write 5 withdraw() methods and Java will know which one you meant — just like the bank knows how to respond based on what you bring."

⇒ Module 5.5 – Interview Questions & Assignment

Beginner Level

No.	Question	Answer
1	What is a method in Java?	A reusable block of code that performs a specific task.
2	What is the syntax of a method?	returnType methodName(parameters) { ... }
3	How do you call a method in Java?	By writing methodName(); or obj.methodName();
4	What does void mean?	It means the method does **not return any value**.
5	What are method parameters?	Values passed into a method during call.
6	Can methods return values?	Yes, by using a return statement and return type.
7	What is method overloading?	Defining multiple methods with **same name** but different parameters.
8	What is recursion?	When a method calls itself to solve a smaller problem.
9	What is a base condition in recursion?	The stopping point of recursion to prevent infinite calls.
10	Can we use multiple methods in a single class?	Yes, that's standard practice.

Intermediate Level

No.	Question	Answer
1	What is the return type of a method that returns nothing?	void
2	Can two methods have the same name and same parameters?	No. That causes a compile-time error.
3	What is the default return type of main()?	void
4	Can a method return multiple values?	Not directly — use arrays or objects.
5	What is the difference between actual and formal parameters?	Actual: passed during call; Formal: declared in method definition.
6	Can methods be overloaded by changing only return type?	No. Parameter list must change.
7	What is StackOverflowError in recursion?	Happens when recursion goes too deep without a base case.
8	Which memory is used in recursive calls?	Stack Memory
9	What is the advantage of using methods?	Code reusability, readability, and modularity.
10	Can we overload main()?	Yes, but JVM always calls public static void main(String[] args)

Expert Level

No.	Question	Answer
1	What is tail recursion?	A recursion where the **last action is the recursive call**, allowing optimization.
2	When is recursion preferred over iteration?	When problem is naturally recursive (e.g., trees, backtracking).
3	Can we write infinite recursion?	Yes — if there's no base condition, it never stops.
4	How is recursion stored in memory?	Every call is stored as a frame in **call stack**.
5	Is recursion faster than iteration?	Usually slower and uses more memory.
6	How does Java resolve overloaded methods?	Based on **method signature** (name + parameter types/number/order).
7	Can you overload a method based on access	No, modifier doesn't matter — only parameter list does.

	modifier only?	
8	◆ What is method signature?	◆ Name + parameter types/number/order (not return type).
9	◆ What's the difference between recursion and backtracking?	◆ Recursion = call itself; Backtracking = **try → backtrack if fail**
10	◆ How would you optimize Fibonacci recursion?	◆ Use **memoization** or **dynamic programming**.

Assignments & Practice for Students

Task	Description
1	Write methods to perform add, subtract, multiply, divide.
2	Create method isPrime(int n) using return type.
3	Overload a method greet() for no parameter, 1 name, 2 names.
4	Write recursive method for **factorial**, **Fibonacci**, **sum of digits**.
5	Create a calculator using method overloading (int, double, float).
6	Write method to reverse a number using recursion.
7	Write method to count vowels in a string.
8	Recursive method to calculate power of a number a^b.
9	Recursive method to print numbers from N to 1.